大規模な集合分割問題に対する局所探索法 (最適化の基礎理論と応用)

AUTHOR(S):

梅谷, 俊治

CITATION:

梅谷, 俊治. 大規模な集合分割問題に対する局所探索法 (最適化の基礎理論と応用). 数理解析研究所講究録 2014, 1879: 91-96

ISSUE DATE:

2014-04

URL:

http://hdl.handle.net/2433/195621

数理解析研究所講究録
第1879巻 2014年 91-96

大規模な集合分割問題に対する局所探索法

梅谷 俊治 (大阪大学)

概 要

集合分割問題は，与えられた集合の全ての要素をちょうど1回ずつ含む部分集合の組み合わせの中で，コストの総和が最小となるものを求める組合せ最適化問題であり，乗務員スケジューリング，配送計画，施設配置など多くの現実問題を応用に持つ．集合分割問題は制約条件を満たす解を求めることも難しく，緩和問題を利用した手法で良い解が得られない場合も少なくないため，NP困難な組合せ最適化問題の中でも特に計算困難な問題として知られている．本研究では，大規模な集合分割問題に対して精度の良い近似解を効率良く求める局所探索法を提案する．提案手法では，同じ制約条件に同時に現れる頻度の高い変数の組を列挙してリストに記憶することで，複数の変数の値を同時に反転する大きな近傍の効率的な探索を実現し，約227万変数の大規模な問題例でも精度の高い近似解が得られることを数値実験で示した．

1 はじめに

集合分割問題は，m 個の要素からなる集合 $M=\{1,2,\ldots,m\}$，n 個の部分集合 $S_j \subseteq M$ $(j \in N=\{1,2,\ldots,n\})$ とそれらのコスト c_j (>0) が与えられたとき，集合 M の全ての要素 $i \in M$ をちょうど1回ずつ含む部分集合の組み合わせの中で，コストの総和が最小となるものを求める問題である．係数 a_{ij} を要素 i が部分集合 S_j に含まれていれば $a_{ij}=1$，含まれていなければ $a_{ij}=0$ と定義し，変数 x_j を部分集合 S_j を選ぶときに $x_j=1$，選ばないときに $x_j=0$ を取る0-1変数として，集合分割問題を以下の0-1整数計画問題に定式化できる．

$$\begin{aligned} \text{min.} \quad & z(\boldsymbol{x})=\sum_{j\in N} c_j x_j & \\ \text{s.t.} \quad & \sum_{j\in N} a_{ij}x_j = 1, & i\in M, \\ & x_j \in \{0,1\}, & j\in N. \end{aligned} \tag{1}$$

集合分割問題は，乗務員スケジューリング，配送計画，施設配置など多くの現実的な問題を応用に持ち，評価法や列生成法を用いた厳密解法や近似解法が提案されている [2, 3, 4, 5, 7, 9]．しかし，原問題と緩和問題の最適値のギャップが大きく，これらの解法では厳密な最適解はおろか精度の良い近似解すら求められない問題例が多く存在することも知られている．これらの問題例では，評価法や列生成法によって高い優先度が付けられているにも関わらず，同じ制約条件に含まれるために同時に選べない変数の組が頻繁に現れることが知られている．

本研究では，大規模な集合分割問題に対して現実的な計算時間で精度の良い近似解を求める局所探索法を提案する．集合分割問題に対して精度の良い近似解を求めるためには，複数の変数の値を同時に反転する大きな近傍を持つ局所探索法を開発することが望ましい．しかし，大規模な集合分割問題では，高々2つの変数の値を同時に反転させる2反転近傍であっても近傍内の候補解の数が非常に多くなるため，大きな近傍を持つ局所探索法では近傍探索の効率化が必要となる．本研究では，同じ制約条件に同時に現れる頻度の高い変数の組を列挙してリストに記憶し，局所探索法では2反転近傍の候補解をこのリストに含まれる変数の組に絞り込むことで探索の効率化を実現する．

2　2反転近傍に基づく局所探索法

局所探索法は，適当な初期解 $\boldsymbol{x}$ から始めて，現在の解 $\boldsymbol{x}$ に少しの変形を加えて得られる解の集合 $\mathrm{NB}(\boldsymbol{x})$ 内に改善解 $\boldsymbol{x}'$ があれば，現在の解 $\boldsymbol{x}$ から $\boldsymbol{x}'$ に移動する操作を反復する方法である．ここで，現在の解 $\boldsymbol{x}$ に変形を加える操作を近傍操作と呼び，それにより生成され得る解の集合 $\mathrm{NB}(\boldsymbol{x})$ を近傍と呼ぶ．現在の解 $\boldsymbol{x}$ の近傍 $\mathrm{NB}(\boldsymbol{x})$ 内に改善解がなければ局所探索法は終了し，このとき得られる解を局所最適解と呼ぶ．

集合分割問題はNP困難のクラスに属する問題であり，さらに制約条件を満たす実行可能解 $\boldsymbol{x}$ が存在するかどうかを判定する問題もNP完全であることが知られている．そこで，全ての制約条件を緩和して0-1ベクトル全体の集合 $\{0,1\}^n$ を探索空間とし，実行不可能解に対しては各制約条件 i の超過と不足の違反度を表すペナルティ y_i^+,y_i^- に重み w_i^+,w_i^- をかけて目的関数に加えるペナルティ関数法に基づく以下の定式化を考える．

$$\begin{aligned} \text{min.} \quad & \tilde{z}(\boldsymbol{x}) = \sum_{j\in N} c_j x_j + \sum_{i\in M} w_i^+ y_i^+ + \sum_{i\in M} w_i^- y_i^- & & \\ \text{s.t.} \quad & \sum_{j\in N} a_{ij} x_j - y_i^+ + y_i^- = 1, & & i \in M, \\ & x_j \in \{0,1\}, & & j \in N, \\ & y_i^+, y_i^- \geq 0, & & i \in M. \end{aligned} \tag{2}$$

ここで，与えられた解 $\boldsymbol{x} \in \{0,1\}^n$ に対して最適な y_i^+, y_i^- は，$y_i^+(\boldsymbol{x}) = \max\{\sum_{j\in N} a_{ij}x_j - 1, 0\}$, $y_i^-(\boldsymbol{x}) = \max\{1 - \sum_{j\in N} a_{ij}x_j, 0\}$ と計算できる．局所探索法では $\tilde{z}(\boldsymbol{x})$ を解の評価関数として，近傍内に $\tilde{z}(\boldsymbol{x})$ の値がより小さい解があれば，それに移動する操作を反復する．このとき，得られる局所最適解は実行可能解とは限らないので，現在の解 $\boldsymbol{x}$ とは別に (実際に訪れた解だけではなく) 探索中に評価した全ての近傍解の中で最良の実行可能解を記憶しておき，これを局所探索法の出力とする．

ペナルティ重み w_i^+, w_i^- が十分に大きければ実行可能解が得られ易くなるが，目的関数値の小さな実行可能解を得るためにはペナルティ重みはあまり大き過ぎない方が良い傾向にある．しかし，ペナルティ重み w_i^+, w_i^- の適切な値を決定することは容易ではなく，適当な値を与えて局所探索法を1回実行するだけでは良い近似解はなかなか得られない．そこで，本研究ではペナルティ重み w_i^+, w_i^- の更新と局所探索法を交互に繰返し適用する方法を用いる．これまでの探索で得られた最良の実行可能解 (暫定解) を $\boldsymbol{x}^*$ とする．局所探索法を終了した時点で得られた局所最適解 $\boldsymbol{x}$ が $\tilde{z}(\boldsymbol{x}) \geq z(\boldsymbol{x}^*)$ を満たすならば，ペナルティ重み w_i^+, w_i^- の値を $\alpha(0 < \alpha < 1)$ 倍して減少させる．そうでなければ，以下の式に従ってペナルティ重み w_i^+, w_i^- の値を増加させる．

$$\begin{aligned} w_i^+ &\leftarrow w_i^+ + \frac{z(\boldsymbol{x}) - \tilde{z}(\boldsymbol{x})}{\sum_{i\in M}(y_i^{+2} + y_i^{-2})} \cdot y_i^+ \quad i \in M, \\ w_i^- &\leftarrow w_i^- + \frac{z(\boldsymbol{x}) - \tilde{z}(\boldsymbol{x})}{\sum_{i\in M}(y_i^{+2} + y_i^{-2})} \cdot y_i^- \quad i \in M. \end{aligned} \tag{3}$$

まず，現在の解 $\boldsymbol{x}$ に対して高々1つの変数の値を反転させて得られる解の集合を近傍とする1反転近傍を探索する．1反転近傍には，$x_j = 0 \to 1$ と反転して得られる解と，$x_j = 1 \to 0$ として得られる解の2種類があり，前者のみからなる近傍を追加近傍，後者のみからなる近傍を削除近傍と呼ぶ．1反転近傍に基づく局所探索法ではまず追加近傍を探索する．現在の解 $\boldsymbol{x}$ からある変数 x_j の値を $x_j = 0 \to 1$ と反転して得られる近傍解 $\boldsymbol{x}'$ と $\boldsymbol{x}$ との評価関数値の差を $\Delta\tilde{z}_j^{\uparrow}(\boldsymbol{x}) = \tilde{z}(\boldsymbol{x}') - \tilde{z}(\boldsymbol{x})$ と定義し，$x_j = 1 \to 0$ と反転する場合についても同様に $\Delta\tilde{z}_j^{\downarrow}(\boldsymbol{x})$ と定義する．追加近傍に $\Delta\tilde{z}_j^{\uparrow}(\boldsymbol{x}) < 0$,

$x_j = 0$ を満たす j が存在すれば $\Delta\tilde{z}_j^{\uparrow}(\boldsymbol{x})$ が最小の j に対して $x_j = 0 \to 1$ と反転して再び追加近傍を探索し，存在しなければ削除近傍に移る．削除近傍も追加近傍と同様に $\Delta\tilde{z}_j^{\downarrow}(\boldsymbol{x}) < 0, x_j = 1$ を満たす j が存在すれば $\Delta\tilde{z}_j^{\downarrow}(\boldsymbol{x})$ が最小の j に対して $x_j = 1 \to 0$ と反転して再び削除近傍を探索し，存在しなければ探索を終了する．

局所探索法では，近傍操作によってごく少数の変数のみ値が変化するため，現在の解 $\boldsymbol{x}$ と近傍解 $\boldsymbol{x}' \in \mathrm{NB}(\boldsymbol{x})$ の間で値が変化した変数に関わる部分のみ再計算すれば評価関数値の変化量 $\Delta\tilde{z}_j^{\uparrow}(\boldsymbol{x})$, $\Delta\tilde{z}_j^{\downarrow}(\boldsymbol{x})$ を高速に計算できる場合が多い．これらは，制約条件 i の左辺値 $s_i(\boldsymbol{x}) = \sum_{j\in N} a_{ij}x_j$ を用いて，

$$\begin{aligned}
\Delta\tilde{z}_j^{\uparrow}(\boldsymbol{x}) &= c_j + \Delta p_j^{\uparrow}(\boldsymbol{x}) + \Delta q_j^{\uparrow}(\boldsymbol{x}),\\
\Delta\tilde{z}_j^{\downarrow}(\boldsymbol{x}) &= -c_j + \Delta p_j^{\downarrow}(\boldsymbol{x}) + \Delta q_j^{\downarrow}(\boldsymbol{x}),\\
\Delta p_j^{\uparrow}(\boldsymbol{x}) &= \sum_{i\in S_j} w_i^{+}\left(\max\{(s_i(\boldsymbol{x})+1)-1,0\} - y_i^{+}(\boldsymbol{x})\right),\\
\Delta q_j^{\uparrow}(\boldsymbol{x}) &= \sum_{i\in S_j} w_i^{-}\left(\max\{1-(s_i(\boldsymbol{x})+1),0\} - y_i^{-}(\boldsymbol{x})\right),\\
\Delta p_j^{\downarrow}(\boldsymbol{x}) &= \sum_{i\in S_j} w_i^{+}\left(\max\{(s_i(\boldsymbol{x})-1)-1,0\} - y_i^{+}(\boldsymbol{x})\right),\\
\Delta q_j^{\downarrow}(\boldsymbol{x}) &= \sum_{i\in S_j} w_i^{-}\left(\max\{1-(s_i(\boldsymbol{x})-1),0\} - y_i^{-}(\boldsymbol{x})\right),
\end{aligned} \tag{4}$$

と計算できる．ここで，制約条件 i の左辺値 $s_i(\boldsymbol{x})$ をあらかじめ計算して補助記憶に持てば，各 $j \in N$ に対する評価関数値の変化量 $\Delta\tilde{z}_j^{\uparrow}(\boldsymbol{x})$, $\Delta\tilde{z}_j^{\downarrow}(\boldsymbol{x})$ を $\mathrm{O}(|S_j|)$ 時間で計算できる．また，$x_j = 0 \to 1$ と反転して現在の解 $\boldsymbol{x}$ から $\boldsymbol{x}'$ に移動した際には，各制約条件 $i \in S_j$ に対して $s_i(\boldsymbol{x}') = s_i(\boldsymbol{x}) + 1$ と計算すれば，補助記憶も $\mathrm{O}(|S_j|)$ 時間で更新できる．$x_j = 1 \to 0$ と反転する場合も同様に各制約条件 $i \in S_j$ に対して $s_i(\boldsymbol{x}') = s_i(\boldsymbol{x}) - 1$ と計算すればよい．

さらに，違反度の変化量 $\Delta p_j^{\uparrow}(\boldsymbol{x})$, $\Delta q_j^{\uparrow}(\boldsymbol{x})$, $\Delta p_j^{\downarrow}(\boldsymbol{x})$, $\Delta q_j^{\downarrow}(\boldsymbol{x})$ そのものを補助記憶に持つことで，各 $j \in N$ に対する評価関数値の変化量 $\Delta\tilde{z}_j^{\uparrow}(\boldsymbol{x})$, $\Delta\tilde{z}_j^{\downarrow}(\boldsymbol{x})$ を $\mathrm{O}(1)$ 時間で計算できる．このとき，$x_j = 0 \to 1$ と反転して現在の解 $\boldsymbol{x}$ から $\boldsymbol{x}'$ に移動した際には，各制約条件 $i \in S_j$ に対して $s_i(\boldsymbol{x}') = s_i(\boldsymbol{x}) + 1$, $y_i^{+}(\boldsymbol{x}') = \max\{s_i(\boldsymbol{x}') - 1, 0\}$, $y_i^{-}(\boldsymbol{x}') = \max\{1 - s_i(\boldsymbol{x}'), 0\}$ と計算した後に，各 $k \in N_i = \{j \in N \mid a_{ij} = 1\}(i \in S_j)$ に対して $\Delta p_k^{\uparrow}(\boldsymbol{x})$, $\Delta q_k^{\uparrow}(\boldsymbol{x})$ を

$$\begin{aligned}
\Delta p_k^{\uparrow}(\boldsymbol{x}') &= \Delta p_k^{\uparrow}(\boldsymbol{x}) + \sum_{i\in S_j\cap S_k} w_i^{+}\left\{\left(\max\{(s_i(\boldsymbol{x}')+1)-1,0\} - y_i^{+}(\boldsymbol{x}')\right) - \left(y_i^{+}(\boldsymbol{x}') - y_i^{+}(\boldsymbol{x})\right)\right\},\\
\Delta q_k^{\uparrow}(\boldsymbol{x}') &= \Delta q_k^{\uparrow}(\boldsymbol{x}) + \sum_{i\in S_j\cap S_k} w_i^{-}\left\{\left(\max\{1-(s_i(\boldsymbol{x}')+1),0\} - y_i^{-}(\boldsymbol{x}')\right) - \left(y_i^{-}(\boldsymbol{x}') - y_i^{-}(\boldsymbol{x})\right)\right\},
\end{aligned} \tag{5}$$

と計算する．$x_j = 1 \to 0$ と反転する場合も同様に，各制約条件 $i \in S_j$ に対して $s_i(\boldsymbol{x}') = s_i(\boldsymbol{x}) - 1$, $y_i^{+}(\boldsymbol{x}') = \max\{s_i(\boldsymbol{x}') - 1, 0\}$, $y_i^{-}(\boldsymbol{x}') = \max\{1 - s_i(\boldsymbol{x}'), 0\}$ と計算した後に，各 $k \in N_i (i \in S_j)$ に対して $\Delta p_k^{\downarrow}(\boldsymbol{x})$, $\Delta q_k^{\downarrow}(\boldsymbol{x})$ を

$$\begin{aligned}
\Delta p_k^{\downarrow}(\boldsymbol{x}') &= \Delta p_k^{\downarrow}(\boldsymbol{x}) + \sum_{i\in S_j\cap S_k} w_i^{+}\left\{\left(\max\{(s_i(\boldsymbol{x}')-1)-1,0\} - y_i^{+}(\boldsymbol{x}')\right) - \left(y_i^{+}(\boldsymbol{x}') - y_i^{+}(\boldsymbol{x})\right)\right\},\\
\Delta q_k^{\downarrow}(\boldsymbol{x}') &= \Delta q_k^{\downarrow}(\boldsymbol{x}) + \sum_{i\in S_j\cap S_k} w_i^{-}\left\{\left(\max\{1-(s_i(\boldsymbol{x}')-1),0\} - y_i^{-}(\boldsymbol{x}')\right) - \left(y_i^{-}(\boldsymbol{x}') - y_i^{-}(\boldsymbol{x})\right)\right\},
\end{aligned} \tag{6}$$

と計算すれば補助記憶は $\mathrm{O}(\sum_{i\in S_j}|N_i|)$ 時間で更新できる．局所探索法では，近傍内の解を評価する回数に比べて現在の解が移動する回数がはるかに少ない場合が多いので，補助記憶の更新に多少時間がかかっても全体では十分な高速化が実現できる．

次に，現在の解 $\boldsymbol{x}$ に対して高々2つの変数の値を反転させて得られる解の集合を近傍とする2反転近傍を探索する．2つの変数 x_{j_1}, x_{j_2} の値を同時に反転させて得られる近傍解 $\boldsymbol{x}'$ と $\boldsymbol{x}$ との評価関数値の差を $\Delta\tilde{z}_{j_1,j_2}(\boldsymbol{x})$ と定義すると，現在の解 $\boldsymbol{x}$ が1反転近傍 $\mathrm{NB}_1(\boldsymbol{x})$ における局所最適解であれば，$x_{j_1} \neq x_{j_2}$ の場合のみ $\Delta\tilde{z}_{j_1,j_2}(\boldsymbol{x}) < 0$ が成り立つ．そこで，2反転近傍では現在の解 $\boldsymbol{x}$ の2つの変数の値を $x_{j_1} = 1 \to 0$, $x_{j_2} = 0 \to 1$ と同時に反転して得られる解のみを考慮すれば良い．このとき，評価関数値の変化量 $\Delta\tilde{z}_{j_1,j_2}(\boldsymbol{x})$ は $\Delta\tilde{z}^{\downarrow}_{j_1}(\boldsymbol{x})$ と $\Delta\tilde{z}^{\uparrow}_{j_2}(\boldsymbol{x})$ を用いて，

$$\Delta\tilde{z}_{j_1,j_2}(\boldsymbol{x}) = \Delta\tilde{z}^{\downarrow}_{j_1}(\boldsymbol{x}) + \Delta\tilde{z}^{\uparrow}_{j_2}(\boldsymbol{x}) - \sum_{i \in S_{j_1} \cap S_{j_2} \cap \{i | s_i(\boldsymbol{x})=1\}} (w_i^+ + w_i^-) \tag{7}$$

と計算できる．1反転近傍で用いた補助記憶を利用すれば補助記憶を更新することなく $\Delta\tilde{z}_{j_1,j_2}(\boldsymbol{x})$ を $O(|S_j|)$ 時間で計算できる．

1反転近傍に改善する解が存在しなければ2反転近傍に移る．しかし，2反転近傍において1反転近傍と同様に $\tilde{z}(\boldsymbol{x}')$ が最小となる解 $\boldsymbol{x}' \in \mathrm{NB}_2(\boldsymbol{x})$ を探索すると多くの計算時間を要する．そこで，2反転近傍 $\mathrm{NB}_2(\boldsymbol{x})$ の中で，$x_j = 1 \to 0$ $(j \in N)$ と反転する候補解の集合を $\mathrm{NB}_2^{(j)}(\boldsymbol{x})$ と定義する．まず，$x_j = 1$ を満たす j を $\Delta\tilde{z}^{\downarrow}_j(\boldsymbol{x})$ の昇順にソートし，この順に $\mathrm{NB}_2^{(j)}(\boldsymbol{x})$ を探索する．$\mathrm{NB}_2^{(j)}(\boldsymbol{x})$ に $\tilde{z}(\boldsymbol{x}') < \tilde{z}(\boldsymbol{x})$ を満たす解 $\boldsymbol{x}'$ が存在すれば，その中で $\tilde{z}(\boldsymbol{x}')$ が最小となる解に移動する．この手続きを2反転近傍の開始時に $x_j = 1$ を満たす全ての $j \in N$ に対して1回ずつ適用する．

3 変数間の関係ネットワークを用いた探索の効率化

集合分割問題に対して精度の良い近似解を求めるためには，複数の変数の値を同時に反転するより大きな近傍を持つ局所探索法を開発することが望ましい．しかし，大規模な集合分割問題では2反転近傍であっても近傍内の候補解の数が非常に多くなるため，大きな近傍を探索する際には改善する見込みの高い候補解にのみ絞り込む必要がある．

前節で示した通り，現在の解 $\boldsymbol{x}$ が1反転近傍における局所最適解であれば，2反転近傍では現在の解 $\boldsymbol{x}$ の2つの変数の値を $x_{j_1} = 1 \to 0$, $x_{j_2} = 0 \to 1$ と同時に反転して得られる解のみを考慮すれば良い．さらに，式(7)から同じ制約条件に同時に現れる頻度の高い変数の組を同時に反転させると改善解が得られ易い傾向があることが分かる．そこで，各変数 x_{j_1} $(j_1 \in N)$ に対して同じ制約条件に現れる変数 $x_{j_2}(S_{j_1} \cap S_{j_2} \neq \emptyset, j_2 \neq j_1)$ のうち頻度 $|S_{j_1} \cap S_{j_2}|$ の高い上位1%を図1(左)に示すリストに格納し，局所探索法では2反転近傍の候補解をこのリストに含まれる変数の組に絞り込むことで探索の効率化を実現する．また，このリストをグラフの隣接リストと見なせば，図1(右)に示す変数間の関係を表すネットワークが得られる．大規模な集合分割問題ではリストの生成に多くの計算時間を要するため，実際には，リストの全ての行を前処理で生成するのではなく，空の状態から始めて，各変数 x_{j_1} を含む2反転近傍を適用する際にリストの変数 x_{j_1} に対応する行を遅延生成する．

4 数値実験

整数計画問題のベンチマーク問題集 [1, 6] に含まれる7つの集合分割問題の問題例に対して，整数計画ソルバーの1つである CPLEX12.5.1 と提案手法 (局所探索法) を MacBookPro(Intel Core i7, 2.7GHz) 上で実行した結果を表1に示す．いずれのアルゴリズムも1スレッドで実行し，計算時間の上限を問題例 air04, air05 では 600 秒に，他の問題例では 3600 秒に設定した．表中の LP 下界値は線形計画緩和問題の最適値を，リスト生成率は提案手法の実行終了時での変数の数に対して生

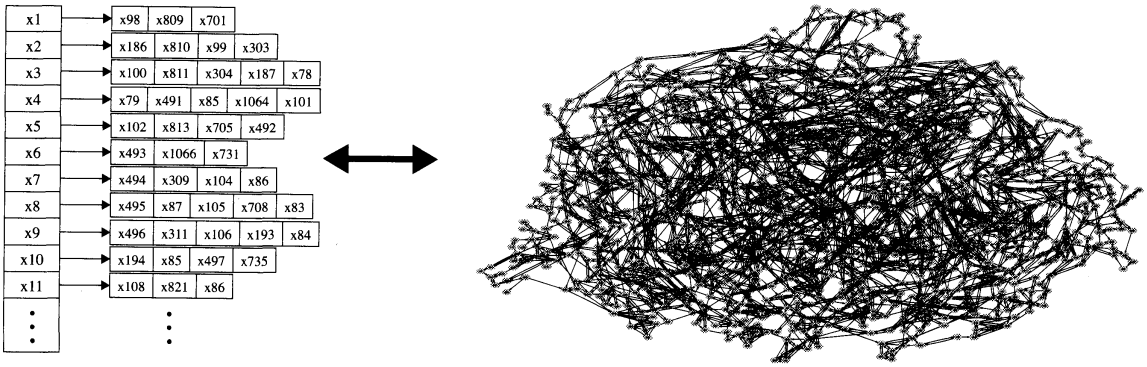

図 1: 同じ制約条件に同時に現れる頻度の高い変数の組を格納したリスト (左) と変数間の関係を表すネットワーク (右)

表 1: MIPLIB の問題例に対する数値実験の結果

問題例	制約	変数	密度	LP 下界値	CPLEX	提案手法	リスト生成率
air04	823	8904	1.00%	55535.43	*56137	60635	62.55%
air05	426	7195	1.70%	25877.60	*26374	27565	65.76%
t1717	551	73,885	0.80%	134531.02	188704	202276	35.12%
t1722	338	36,630	1.08%	98815.40	122532	122371	38.59%
ds	656	67,732	2.31%	57.23	384.25	235.13	34.00%
ds-big	1042	174,997	2.54%	86.82	1771.07	1227.96	5.92%
ivu06-big	1177	2,277,736	0.87%	135.42	672.49	188.58	0.21%

成されたリストの行数の割合を，CPLEX，提案手法はそれぞれ実行終了時における CPLEX12.5.1 と提案手法の暫定解 (実行可能解) の目的関数値を表す．また，アスタリスク (*) の印は厳密な最適値が得られたことを表す．

表 1 から見て取れるように，ds-big や ivu06-big などの大規模な問題例に対して，分枝カット法を基本戦略とする整数計画ソルバーでは限られた時間内に精度の良い近似解を求めることは容易ではない．ちなみに，問題例 ds では大規模な計算機上で整数計画ソルバーを長時間並列実行することで最適値 93.52 が得られている [8]．提案手法では，2 反転近傍の候補解となる変数の組を 1%に絞り込んでいる上に，ds-big や ivu06-big などの大規模な問題例では生成されたリストの行数の割合が非常に低いことから，2 反転近傍のごくわずかな候補解を探索するだけで精度の良い近似解を求めていることが分かる．air04 や air05 などの小規模な問題例では，整数計画ソルバーでは厳密な最適解が求まる一方で，提案手法では最適解に近い解を求めることも容易ではない．また，実行可能解を 1 つ求めること自体が非常に難しい問題例も多く，アルゴリズムの仕様やパラメータの値を少し変更するだけで結果が大きく変わる．例えば，問題例 ds-big, ivu06-big では提案手法を実装する際の試行錯誤でそれぞれ 1053.06, 176.61 の暫定値が得られており，まだ改善の余地は大きいと予想される．

5 おわりに

本研究では，大規模な集合分割問題に対して精度の良い近似解を効率良く求める局所探索法を提案した．集合分割問題に対して精度の良い近似解を求めるためには，複数の変数の値を同時に

反転する大きな近傍を持つ局所探索法を開発することが望ましい．しかし，大規模な集合分割問題では2反転近傍であっても近傍内の候補解が非常に多くなるため，同じ制約条件に同時に現れる頻度の高い変数の組をリストに記憶し，局所探索法では2反転近傍の候補解をこのリストに含まれる変数の組に絞り込むことで探索の効率化を実現した．また，リストの全ての行を前処理するのではなく，2反転近傍の探索時に必要な行のみを遅延生成することで，大規模な集合分割問題に対して現実的な計算時間で精度の高い近似解を求めることを可能とした．

参考文献

[1] T. Achterberg, T. Koch and A. Martin, MIPLIB2003, *Operations Research Letters*, **34** (2006), 361-372.

[2] A. Atamtürk, G. L. Nemhauser and M. W. P. Savelsbergh, A combined Lagrangian, linear programming, and implication heuristic for large-scale set partitioning problems, *Journal of Heuristics*, **1** (1995), 247-259.

[3] F. Barahona and R. Anbil, The volume algorithm: producing primal solutions with a subgradient method, *Mathematical Programming*, **A87** (2000), 385-399.

[4] R. Borndörfer, Aspect of set packing, partitioning and covering, Ph. D. Dissertation, Technischen Universität, Berlin, 1998.

[5] M. A. Boschetti, A. Mingozzi and S. Ricciardelli, A dual ascent procedure for the set partitioning problem, *Discrete Optimization*, **5** (2008), 735-747.

[6] T. Koch, T. Achterberg, E. Andersen, O. Bastert, T. Berthold, R. .E. Bixby, E. Danna, G. Gamrath , A. M. Gleixner, S. Heinz, A. Lodi, H. Mittelmann, T. Ralphs, D. Salvagnin, D. E. Steffy and K. Wolter, "MIPLIB2010 Mixed integer programming library version 5," *Mathematical Programming Computation*, **3** (2011), 103-163.

[7] J. T. Linderoth, E. K. Lee and M. W. P. Savelsbergh, A parallel, linear, programming based heuristic for large-scale set partitioning problems, *INFORMS Journal on Computing*, **13** (2001), 191-209.

[8] 品野勇治, T. Achterberg, T. Berthold, S. Heinz, T. Koch, S. Vigerske and M. Winkler, 制約整数計画ソルバSCIPの並列化, 統計数理, **61** (2013), 47-78.

[9] D. Wedelin, An algorithm for large-scale 0-1 integer programming with application to airline crew scheduling, *Annals of Operations Research*, **57** (1995), 283-301.